Amendments to the Claims

This listing of claims will replace all prior versions, and listings, of claims in the application:

1. (Previously Presented) A method comprising:

storing a register architectural state of a processor corresponding to a first checkpoint;

storing non-deterministic memory access events as associated data that occur subsequent to the storage of the first checkpoint;

determining whether an processing error has occurred subsequent to the storage of the first checkpoint; and

restoring the register architectural state of the processor corresponding to the first checkpoint and re-executing the non-deterministic memory access events if a processing error is detected.

2. (Previously Presented) The method of claim 1 wherein storing the register architectural state of the processor corresponding to the first checkpoint comprises:

synchronizing a leading thread of instructions and a trailing thread of instructions;

checking outputs from the leading thread of instructions and the trailing thread of instructions for faults; and

storing values from one or more architectural registers in a memory external to one or more processors executing the leading thread of instructions and the trailing thread of instructions.

3. (Original) The method of claim 2 wherein the leading thread of instructions and the trailing thread of instructions are executed by a single processor.

4. (Original) The method of claim 2 wherein the leading thread of instructions and the trailing thread of instructions are executed by multiple processors.

5. (Previously Presented) The method of claim 1 wherein restoring the register architectural state of the processor corresponding to the first checkpoint and re-executing the non-deterministic memory access events if a processing error is detected comprises:

selectively flushing results of instructions that started execution after an instruction causing the fault started execution;

restoring register architectural state to a checkpoint corresponding to a state at which the instruction causing the fault started execution; and

re-executing instructions executed after the checkpoint to the instruction causing the fault using the logged non-deterministic memory access events.

6. (Previously Presented) The method of claim 5 further comprising discarding one or more outputs generated by the re-execution of the instructions executed

after the checkpoint to the instruction causing the fault using the logged non-deterministic memory access events.

7. (Original) The method of claim 5 further comprising continuing execution of instructions subsequent to the instruction causing the fault.

8. (Previously Presented) The method of claim 5 wherein restoring the register architectural state to a checkpoint corresponding to a state at which the instruction causing the fault started execution comprises restoring the architectural state to a state prior to execution of the instruction causing the fault.

9. (Previously Presented) The method of claim 5 wherein restoring the register architectural state to a checkpoint corresponding to a state at which the instruction causing the fault started execution comprises restoring the register architectural state to a state a the beginning of execution of the instruction causing the fault.

10. (Previously Presented) The method of claim 1 wherein the non-deterministic memory access event comprises a load operation.

11. (Previously Presented) The method of claim 1 wherein the non-deterministic memory access event comprises an interrupt.

12. (Original) The method of claim 11 wherein the interrupt comprises an asynchronous interrupt.

13. (Previously Presented) The method of claim 1 wherein the non-deterministic memory access event comprises a timing-dependent operation.

14. (Original) The method of claim 13 wherein the timing-dependent operation comprises a read operation of a cycle counter.

15. (Previously Presented) An apparatus comprising:

means for storing a register architectural state of a processor corresponding to a first checkpoint;

means for storing non-deterministic memory access events as associated data that occur subsequent to the storage of the first checkpoint;

means for determining whether an processing error has occurred subsequent to the storage of the first checkpoint; and

means for restoring the register architectural state of the processor corresponding to the first checkpoint and re-executing the non-deterministic memory access events if a processing error is detected.

16. (Previously Presented) The apparatus of claim 15 wherein the means for restoring the register architectural state of the processor corresponding to the first

checkpoint and re-executing the non-deterministic memory access events if a processing error is detected comprises:

means for selectively flushing results of instructions that started execution after an instruction causing the fault started execution;

means for restoring the register architectural state to a checkpoint corresponding to a state at which the instruction causing the fault started execution; and

means for re-executing instructions executed after the checkpoint to the instruction causing the fault using the logged non-deterministic events.

17. (Previously Presented) The apparatus of claim 15 further comprising means for discarding one or more outputs generated by the re-execution of the instructions executed after the checkpoint to the instruction causing the fault using the logged non-deterministic memory access events.

18. (Previously Presented) An apparatus comprising:

leading thread execution circuitry to execute a leading thread of instructions;

trailing thread execution circuitry to execute a trailing thread of instructions; and

a memory coupled with the leading thread execution circuitry and the trailing thread execution circuitry to store information related to non-deterministic memory access events, wherein the information related to non-deterministic memory access events is stored at least until a subsequent checkpoint having an associated register architectural state is validated.

19. (Original) The apparatus of claim 18 wherein the memory stores a load value queue that stores replication entries corresponding to load operations executed by the leading thread execution circuitry and not executed by the trailing thread execution circuitry, and further wherein the load value queue stores recovery entries corresponding to load operations executed by the leading thread execution circuitry and by the trailing thread execution circuitry and not incorporated into information corresponding to a checkpoint.

20. (Original) The apparatus of claim 19 wherein the load value queue further stores information corresponding to one or more of an interrupt and a timing-dependent operation.

21. (Original) The apparatus of claim 18 wherein the memory stores a first load value queue that stores replication entries corresponding to load operations executed by the leading thread execution circuitry and not executed by the trailing thread execution circuitry, and a second load value queue stores recovery entries corresponding to load operations executed by the leading thread execution circuitry and by the trailing thread execution circuitry and not incorporated into information corresponding to a checkpoint.

22. (Original) The apparatus of claim 21 wherein the first load value queue and the second load value queue further store information corresponding to one or more of an interrupt and a timing-dependent operation.

23. (Currently Amended) A system comprising:

leading thread execution circuitry to execute a leading thread of instructions;

trailing thread execution circuitry to execute a trailing thread of instructions;

a memory controller coupled with the leading thread execution circuitry; and

a memory coupled with the leading thread execution circuitry and the trailing thread execution circuitry to store information related to non-deterministic memory access events, wherein ~~the information related to the non-deterministic memory access events is stored at least until a subsequent checkpoint having an associated register architectural state is validated and the non-deterministic memory access events are re-executed if a processing error is detected~~ the memory controller causes the memory to store a register architectural state of a processor corresponding to a first checkpoint and non-deterministic memory access events as associated data that occur subsequent to the storage of the first checkpoint, the memory controller determines whether an processing error has occurred subsequent to the storage of the first checkpoint, and, in response to the processing error, the memory controller restores the register architectural state of the processor corresponding to the first checkpoint and re-executes the non-deterministic memory access events.

24. (Original) The system of claim 23 wherein the memory stores a load value queue that stores replication entries corresponding to load operations executed by the leading thread execution circuitry and not executed by the trailing thread execution circuitry, and further wherein the load value queue stores recovery entries corresponding to load operations executed by the leading thread execution circuitry and by the trailing

thread execution circuitry and not incorporated into information corresponding to a checkpoint.

25. (Original) The system of claim 24 wherein the load value queue further stores information corresponding to one or more of an interrupt and a timing-dependent operation.

26. (Original) The system of claim 23 wherein the memory stores a first load value queue that stores replication entries corresponding to load operations executed by the leading thread execution circuitry and not executed by the trailing thread execution circuitry, and a second load value queue stores recovery entries corresponding to load operations executed by the leading thread execution circuitry and by the trailing thread execution circuitry and not incorporated into information corresponding to a checkpoint.

27. (Original) The system of claim 26 wherein the first load value queue and the second load value queue further store information corresponding to one or more of an interrupt and a timing-dependent operation.